



Creación de Aplicaciones Python

Lección 1: Conocimientos mínimos generales [1/3]

ÍNDICE

Lección 1. – Conocimientos mínimos generales [1/3] 2

Presentación y objetivos.....	2
1. Python y su potencial en desarrollo de software.....	5
2. Explicación DEL CONTENIDO del curso.....	9
3. Guías de instalación del entorno de desarrollo ATOM en Windows.....	12
4. Guías de instalación ATOM en LINUX UBUNTU 20.04.....	14
5. Configuración de ATOM e Instalación de “packages”	17
6. Guía de instalación Postman para WINDOWS.....	24
7. Guía de instalación Postman para LINUX UBUNTU 20.04	27
8. Guía de instalación entornos virtuales para WINDOWS	29
9. Guía de instalación Entornos Virtuales para LINUX UBUNTU 20.04	34
10. Anaconda: No necesaria instalación pero utilizado puntualmente.....	38
11. Guía de instalación Qt Designer en Windows.....	39
12. Guía de instalación Qt Designer LINUX UBUNTU 20.04.....	42
13. Puntos clave.....	44

Lección 1. – Conocimientos mínimos generales [1/3]

PRESENTACIÓN Y OBJETIVOS



tkinter — Python interface to Tcl/Tk

PyQt

PyWebIO



Fuentes de obtención de los Logos:

<https://www.python.org/community/logos/>

<https://www.w3.org/html/logo/>

<https://www.vectorlogo.zone/>

<https://docs.python.org/3/library/tkinter.html>

<https://www.riverbankcomputing.com/software/pyqt/>

<https://www.django-rest-framework.org/>

<https://www.djangoproject.com/community/logos/>

<https://github.com/scikit-learn/scikit-learn/tree/main/doc/logos>

<https://pandas.pydata.org/about/citing.html>

<https://numpy.org/>

<https://github.com/numpy/numpy/blob/main/branding/logo/primary/numpylogo.png>

https://commons.wikimedia.org/wiki/File:Jupyter_logo.svg

<https://streamlit.io/brand>

<https://pywebio.readthedocs.io/en/latest/index.html>

<https://fastapi.tiangolo.com/>

https://commons.wikimedia.org/wiki/File:CSS3_logo_and_wordmark.svg

<https://bokeh.org/branding/>

<https://plotly.com/dash/>

<https://seaborn.pydata.org/citing.html>

https://github.com/mwaskom/seaborn/blob/master/doc/_static/logo-wide-lightbg.png

https://facebook.github.io/prophet/docs/quick_start.html#python-api

<https://pycaret.org/about/>

Haremos Aplicaciones que nos permitan visualizar gráficas.

```
In [31]: data.plot(title = "Gráfica de cotización de Google",  
                  xlabel = "2015-2018",  
                  ylabel = "Cotización en $ USD" )  
  
plt.show()
```



Figura 0.1: Ejemplo de visualización de gráficos.

E incluso aplicaciones con autenticación de usuario, explicando qué papel cumple cada cosa

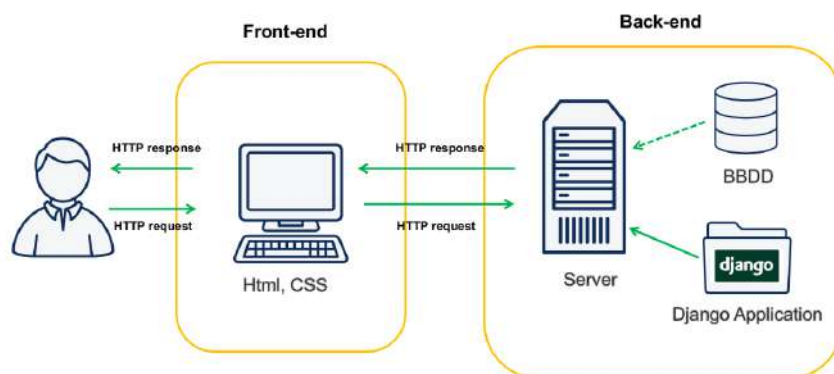


Figura 0.2: Ejemplo de funcionamiento de Django framework

1. PYTHON Y SU POTENCIAL EN DESARROLLO DE SOFTWARE

Si te encuentras aquí es porque has hecho una “apuesta” personal por el Lenguaje de Programación Python. Y, aunque la tecnología cambia muy rápido, y pueden cambiar las cosas, te encuentras en el lugar correcto.

Ya por el año 2015-2016 se podía visualizar el escenario actual, con Python como Lenguaje de Programación más importante y capaz de hacer casi de todo.

Algunos/as programadores iban “saltando” de otros lenguajes de programación a Python. Sin ánimo de crear debate, ya se estaba migrando de R a Python en muchos lugares, (fruto de la existencia de herramientas para Inteligencia Artificial como TensorFlow de Google), incluso de otros Lenguajes de Programación tradicionales como Java a Python.

Ya, en aquel entonces, se podía ver cómo iban saliendo librerías para casi todo.

A día de hoy (actualmente 2021) con una mayor madurez de la tecnología se puede confirmar la predicción. Python puede usarse para proyectos de Inteligencia Artificial, Ciberseguridad, Web, Internet of Things (IoT), etc.

En el caso que nos ocupa, la programación de Aplicaciones Web usando Frameworks con Python, es un claro ejemplo.

Pero, tal vez te estés preguntando.. ¿qué es un Framework? ¿necesito programar JavaScript, HTML, CSS o alguna otra cosa más?

Un Framework es algo que nos va a permitir ahorrar tiempo en la programación, gracias a algo ya creado, ya programado y, depende de qué Framework utilices, y para que finalidad, pero podríamos comentar lo siguiente: HTML y CSS van a seguir siendo necesarios o por lo menos recomendables en algunos de los Frameworks, aunque el tiempo de desarrollo va a ser infinitamente menor.

Para esta asignatura estaremos viendo los siguientes Frameworks:

- | **Dash**, ideal para gráficos de cotizaciones de bolsa, por ejemplo.
- | **Python Django**, un Framework Full-Stack (**REST Framework** incluido)
- | **Flask**, muy popular aunque a día de hoy le está saliendo competencia
- | **FastAPI**, algo que está ganando popularidad por sus ventajas.
- | **Streamlit**, algo que está empezando a revolucionar el mundo de las aplicaciones de Inteligencia Artificial
- | **TKinter**, para Aplicaciones GUI

- | **PyQT** para Aplicaciones GUI
- | **PyGTK** éste quizá no sea visto. (Dependerá de la marcha del curso).
- | **PyWebIO** – Otra gran apuesta en el entorno de la Inteligencia Artificial

Nota: A día de hoy -abril 2021- y viendo el empuje que lleva Streamlit y que se están migrando cientos de aplicaciones de Flask a Streamlit, (y quizá es solo el principio), da que pensar, no obstante, lo ideal es proporcionar unas bases mínimas en estos Frameworks, y, según las necesidades de cada uno/a pueda elegir en base a la demanda laboral y a los gustos personales.

(También obviamente en base a las potenciales ventajas de cada Framework)

Al mismo tiempo, y puesto que el programa formativo -el propio Máster- está centrado también en Machine Learning, tocaremos conocimientos en este campo -serán parcialmente evaluables, o no evaluables- puesto que existen ya asignaturas de este tipo.

Pero veremos la utilidad de crear Aplicaciones Web para Inteligencia Artificial.

Por ello los conocimientos que se impartirán en este punto serán aquellos que permitan a los/as estudiantes "NO ATASCARSE" de forma innecesaria en las próximas lecciones, y que permitan CONOCER que posibilidades tenemos si compaginamos las mismas con los propios Frameworks de Python.

El potencial se multiplica.

Si bien, como indicaremos, no se explicará todo elecciones, si se tratará de abordar conocimientos esenciales, y necesarios.

Cada uno/a podrá profundizar (si lo necesita) en un futuro, siendo proactivo/a en proyectos personales, laboralmente, etc.



Objetivos

- Conocer los conocimientos necesarios para abordar el resto del curso
- Explicación de los proyectos que van a realizarse y, por ello, el itinerario formativo que se presenta
- Guías de Instalación de ATOM, Postman, Qt Designer y Entornos Virtuales.

Conocimientos a tratar por tanto en esta introducción (primeras 3 clases):

- | HTML
- | CSS
- | PYTHON (Muy breve repaso -cosas útiles en esta lección)
- | Análisis básico de datos con Machine Learning

Obviamente, la parte de obtención de datos y Machine Learning será muy breve, de hecho demasiado breve, pero nos servirá para que podamos abordar las lecciones siguientes sin mayor problema y que nos “vaya sonando” para futuras asignaturas/módulos formativos

Algo que también proporcionaremos será unas Guías de Instalación

Todas las guías tendrán explicación de las mismas para Windows y para Linux.

Explicación de las Guías:

- | IDE: seguiremos ATOM como entorno de desarrollo principal, puesto que vamos a crear archivos con extensión .py y tiene muchas ventajas.
- | Postman: Para comprobación de API REST.
- | Entornos Virtuales, para evitar problemas con las versiones de las librerías
- | Jupyter, aunque no será necesaria su instalación, es ideal para Data Science
- | Qt Designer, necesario para uno de los Frameworks

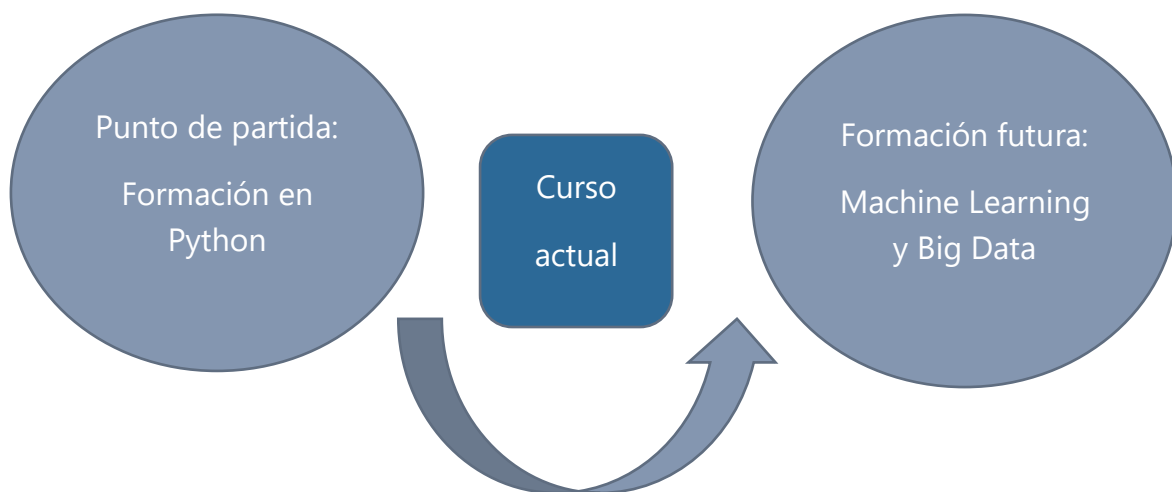
Para uno de los Frameworks comentaremos que es posible implementar ese desarrollo en entornos como Jupyter Notebook (con extensión por defecto: .ipynb), pero éste no será el entorno principal del curso, tiempo tendréis de usar el propio Jupyter, Google Colab o incluso PyCharm.

(De modo que probablemente no lo instalaremos- tampoco por ello el software de "Anaconda", algo recomendable aunque tampoco necesario)

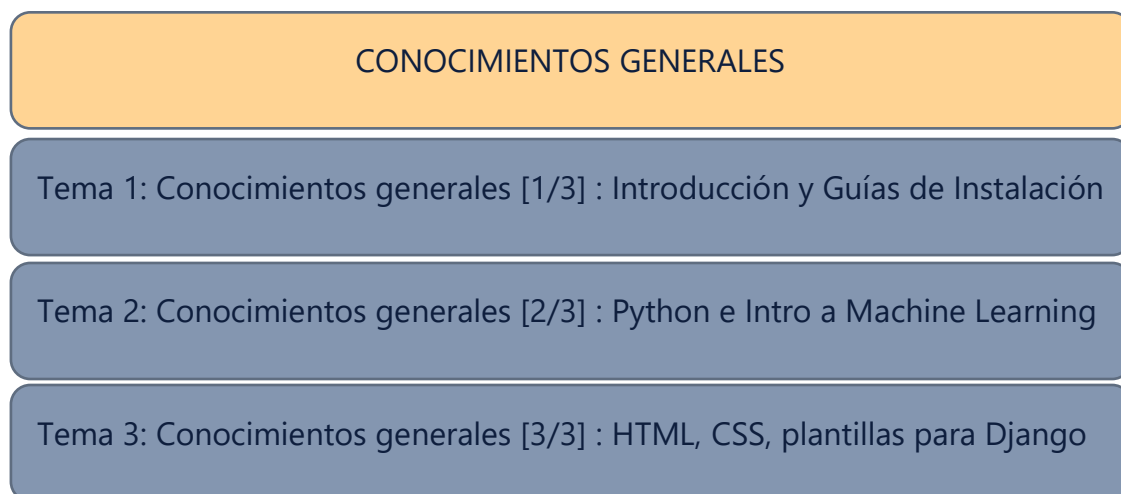
NOTA: Si se prefiere utilizar otro IDE es posible, incluso el propio IDLE de Python.
En nuestro caso estaremos utilizando ATOM por su actual importancia.

2. EXPLICACIÓN DEL CONTENIDO DEL CURSO

Explicación del contenido global del curso y de los motivos de estar así enfocado



Para ello, hemos diseñado el siguiente itinerario formativo:



FRAMEWORK FULL-STACK: DJANGO

Tema 4: Django [1/4] : Configuración Inicial

Tema 5: Django [2/4] : Autenticación de usuarios

Tema 6: Django [3/4] : Rest Framework

Tema 7: Django [4/4] : Traducción



FRAMEWORK POPULAR: FLASK

Tema 8: Flask



FRAMEWORK MODERNO Y RÁPIDO: FastAPI

Tema 9: FastAPI



FRAMEWORKS GUI

Tema 10: Frameworks GUI [1/2] : Tkinter

Tema 11: Frameworks GUI [2/2] : PyQt



FRAMEWORKS PARA GRÁFICAS: DASH

Tema 12: Frameworks para Gráficas: APIs, Series Temporales y el propio Dash



FRAMEWORKS MODERNOS Y CON MUCHO POTENCIAL

Tema 13: Frameworks modernos [1/2]: Streamlit

Tema 14: Frameworks modernos [2/2]: PyWebIO

Se ha tratado de realizar un temario con contenido lo más “balanceado” posible, de tal manera que el/la estudiante, al finalizar esta asignatura disponga de las herramientas necesarias para poder crear proyectos más escalables, y de forma que se entienda la utilidad de cada Framework.

3. GUÍAS DE INSTALACIÓN DEL ENTORNO DE DESARROLLO ATOM EN WINDOWS

Para el Sistema Operativo Windows, nos vamos al siguiente Link:

<https://atom.io/>

Teniendo en cuenta los siguientes requisitos: "For 64-bit Windows 7 or later"

Indicamos "Download" → Elegimos una ruta.

Doble click en el icono:



Nos indicará: "Atom is being installed. It will launch once it is done"

Y al terminar nos quede el siguiente icono, que haciendo doble click sobre él..



De momento decimos que "No" al menú que sale a la derecha.

<https://discuss.atom.io/t/register-as-default-atom-uri-handler/50982/4>

Queda de la siguiente manera:

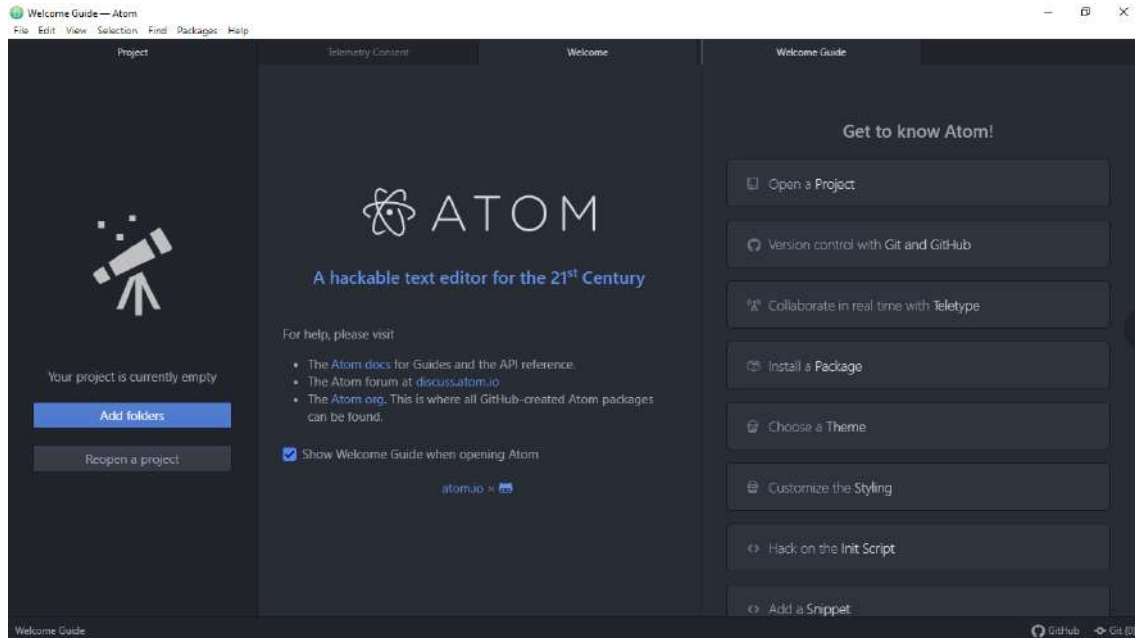


Figura 3.1: Pantalla de inicio de Atom

File → New File

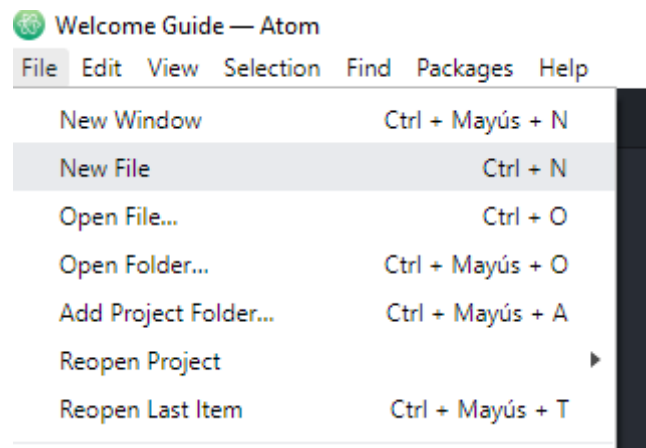


Figura 3.2: Ejemplo de cómo crear un nuevo archivo

Cerramos el resto de ventanas.

Y comenzaríamos a escribir nuestro código.

4. GUÍAS DE INSTALACIÓN ATOM EN LINUX UBUNTU 20.04

Vamos a realizar la guía de instalación de Atom que será el programa que vamos a usar para la creación de Aplicaciones Web, para ello, nosotros usamos un sistema operativo Ubuntu 20.04:

En él encontramos un menú de aplicaciones disponibles:

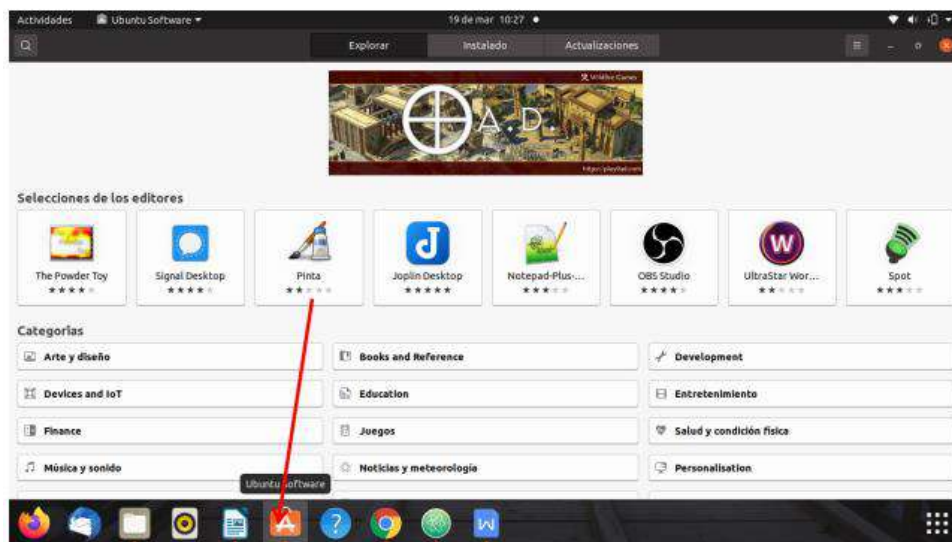


Figura 4.1: Ubuntu software (donde se encuentran las aplicaciones disponibles)

Vamos a la lupa para poder buscar la aplicación de Atom :



Figura 4.2: Búsqueda de Atom en Ubuntu software

Una vez seleccionada damos a instalar

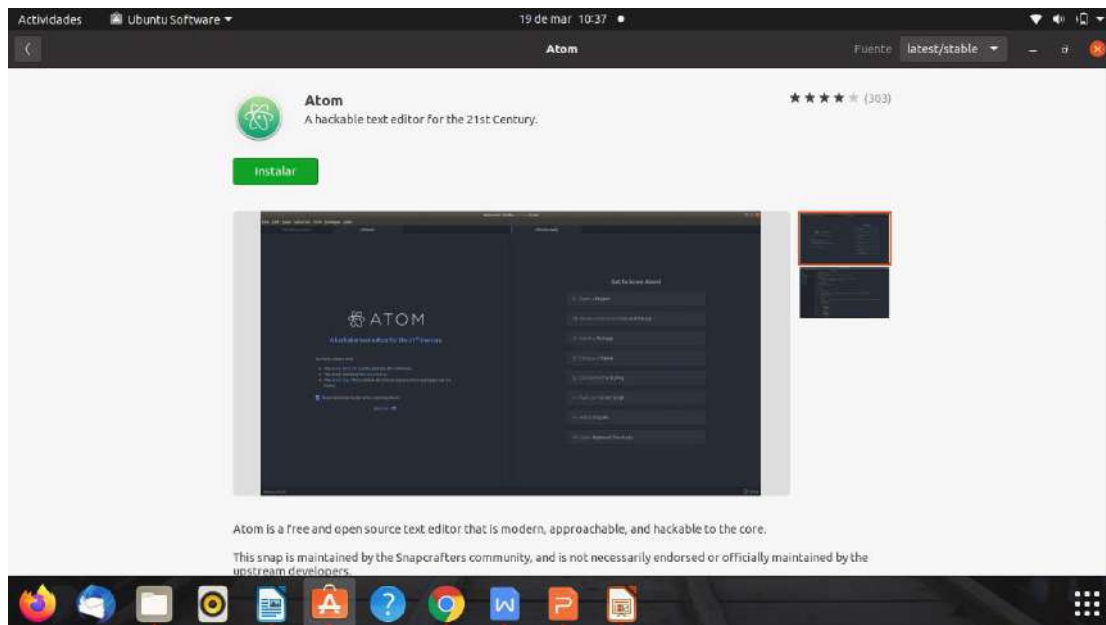


Figura 4.3: Instalación de Atom en Ubuntu software

Una vez instalada iremos a mostrar aplicaciones (esquina inferior derecha) y veremos el logo de Atom, pulsaremos y nos abrirá la aplicación:



Figura 4.4: Menú de aplicaciones disponibles instaladas

Enhorabuena ya tenemos la aplicación instalada:

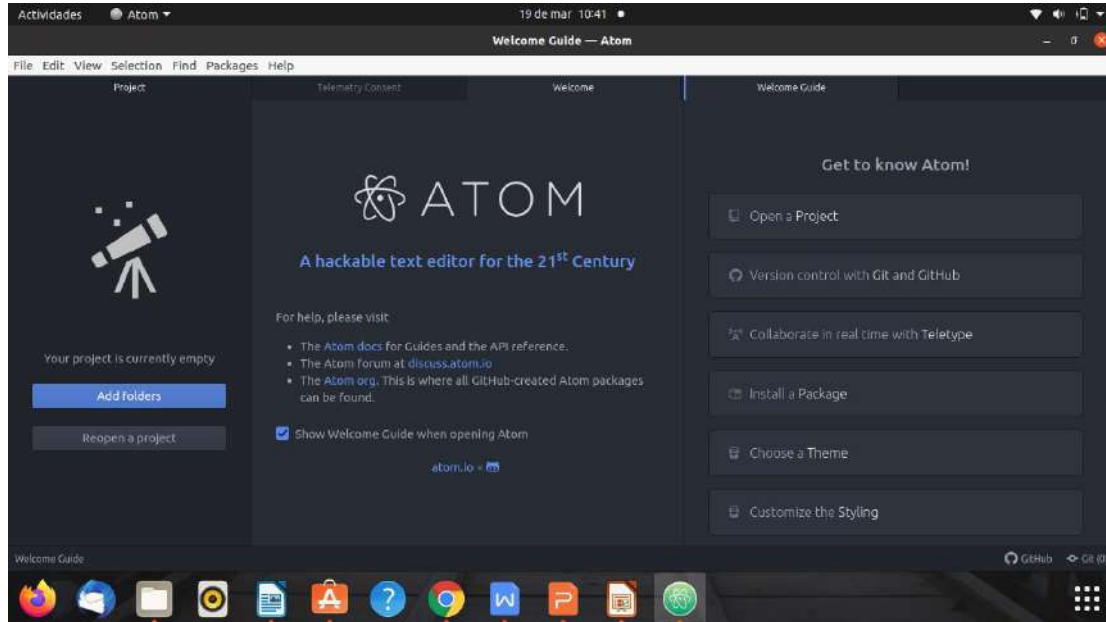


Figura 4.5: Pantalla de inicio Atom

5. CONFIGURACIÓN DE ATOM E INSTALACIÓN DE “PACKAGES”

Una vez instalado ATOM, es aconsejable instalar algunas cosas mas

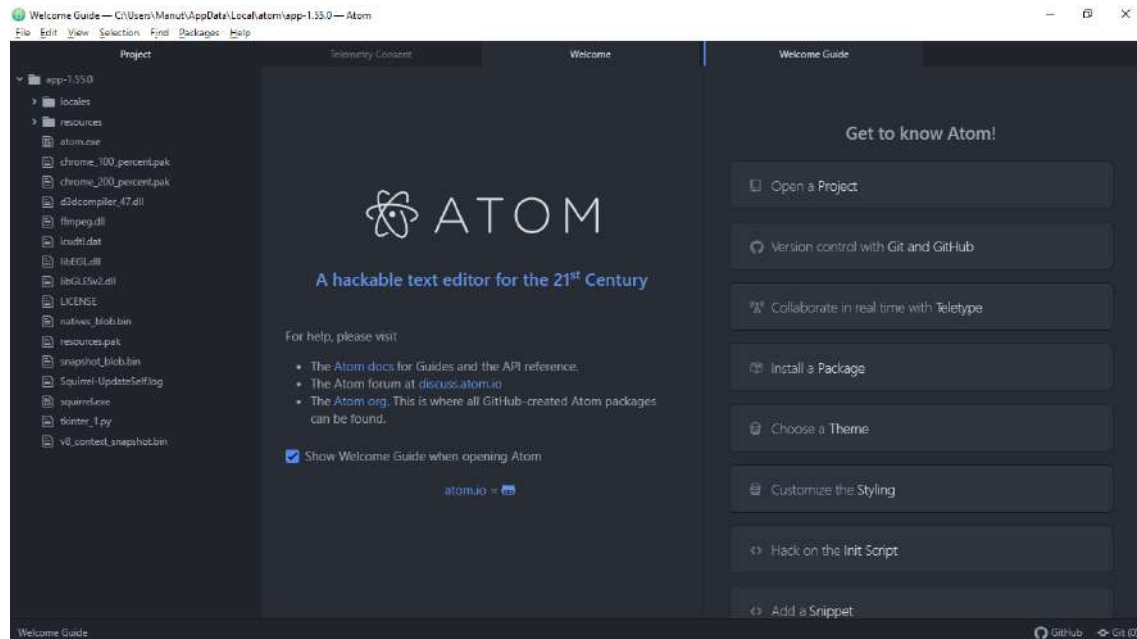


Figura 5.1: Pantalla de inicio Atom para instalación de paquetes

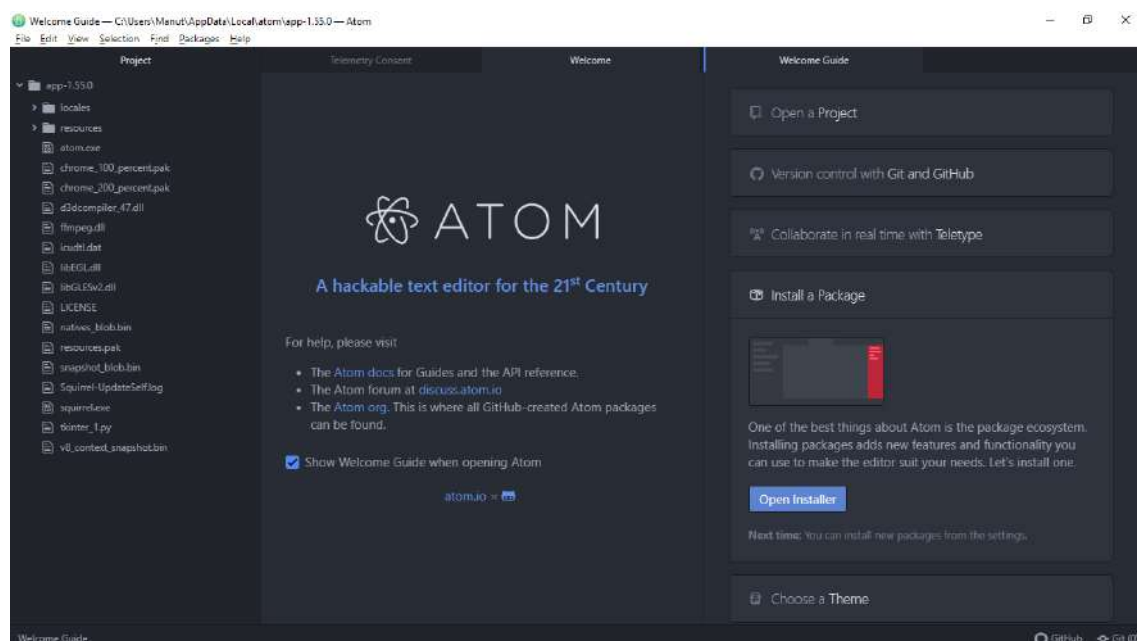


Figura 5.2: Instalación de paquetes en Atom

“script”

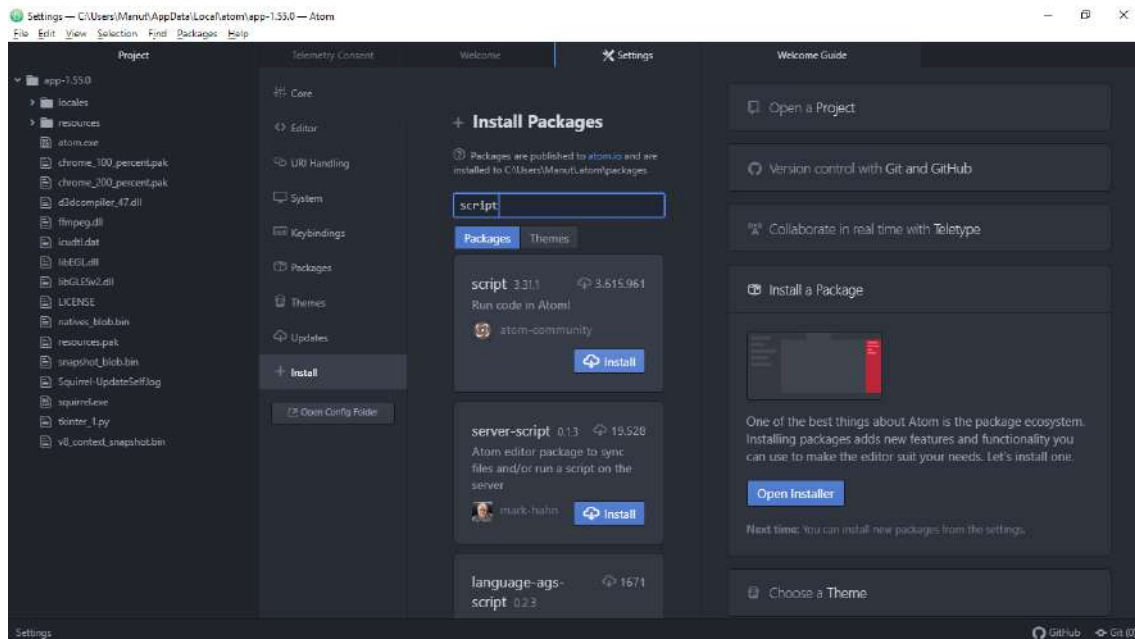


Figura 5.3: Instalación de paquete script en Atom

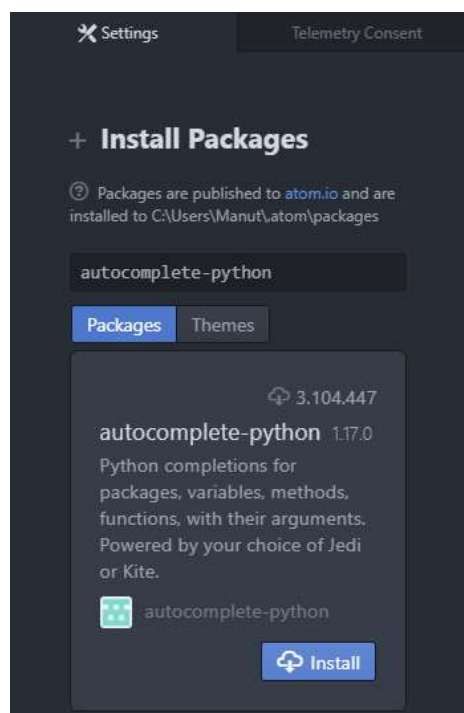


Figura 5.4: Instalación de paquete autocomplete-Python en Atom

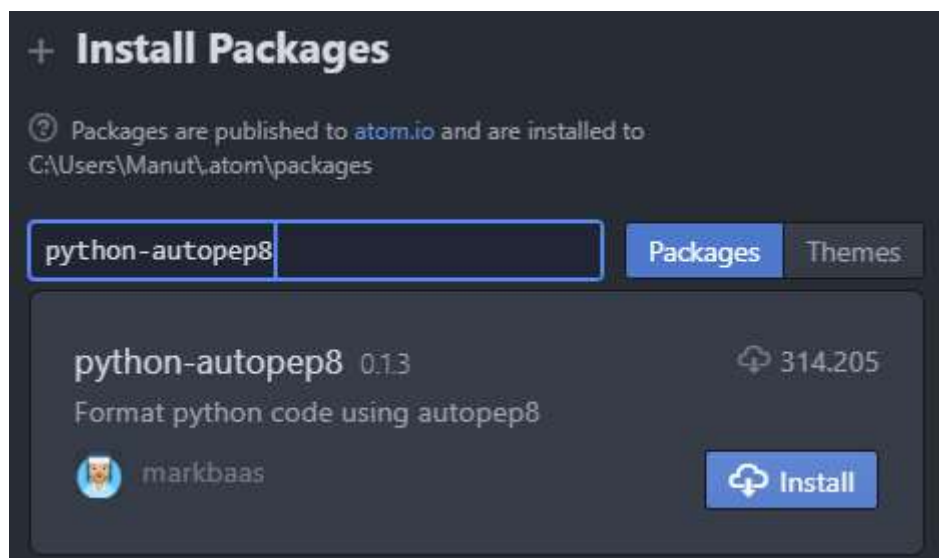


Figura 5.5: Instalación de paquete `python-autopep8` en Atom

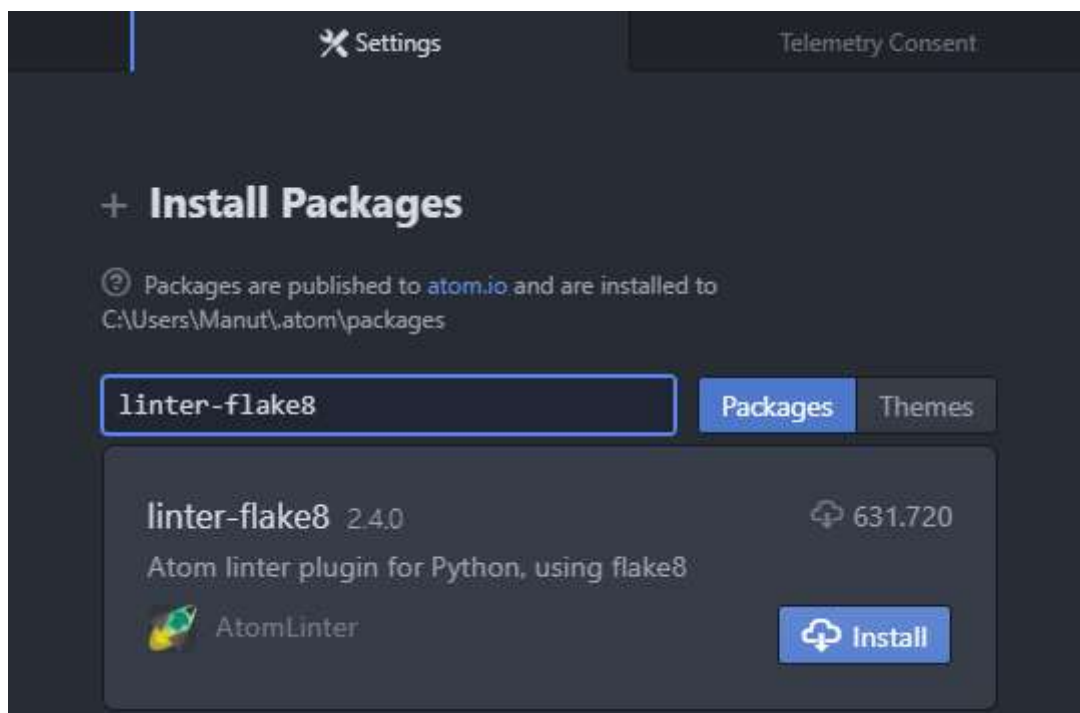


Figura 5.6: Instalación de paquete `linter-flake8` en Atom

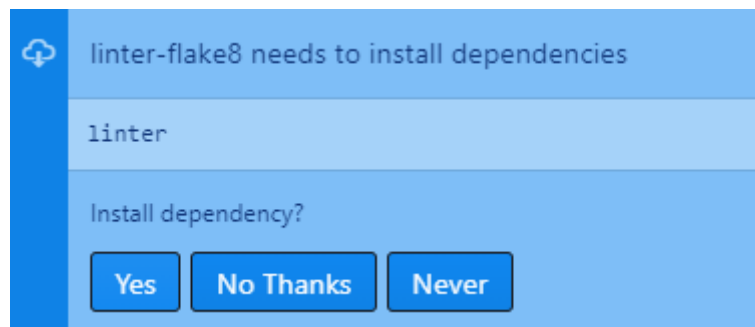


Figura 5.7: Instalación de paquete linter dependencies en Atom

Podría dar el siguiente error

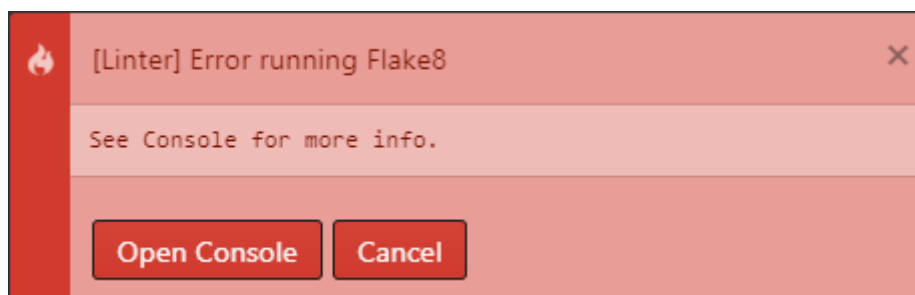


Figura 5.8: Error running Flake8 en Atom

Entonces, la forma de solucionarlo es la siguiente:

Primeramente, desinstalas, de esta forma: "Uninstall"

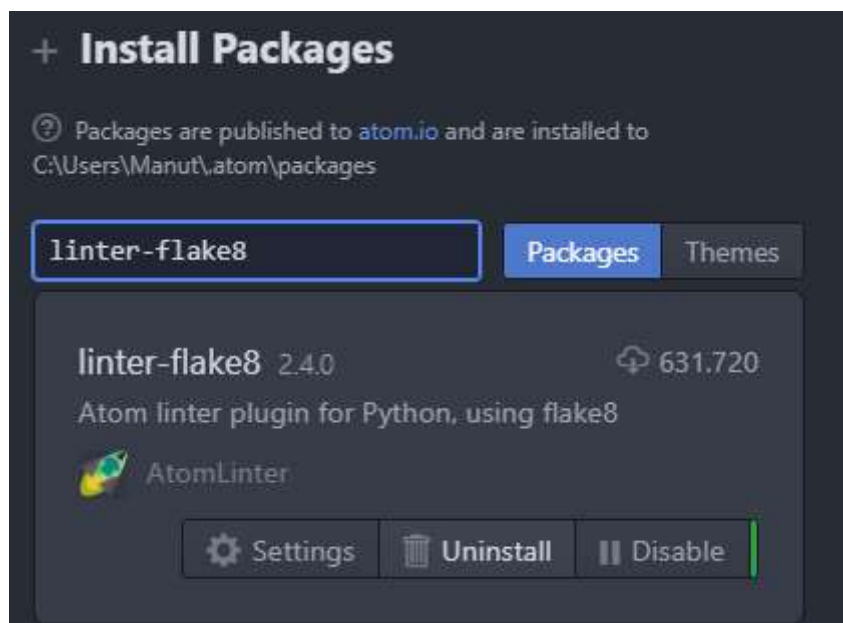


Figura 5.9: Desinstalación linter-flake8 en Atom

Normalmente, suele hacerse: "pip install <nombre> (Ojo, no siempre)

Haciendo una búsqueda rápida en Google: "pip install flake8" nos encontramos con el siguiente link:

<https://pypi.org/project/flake8/>

Y efectivamente,

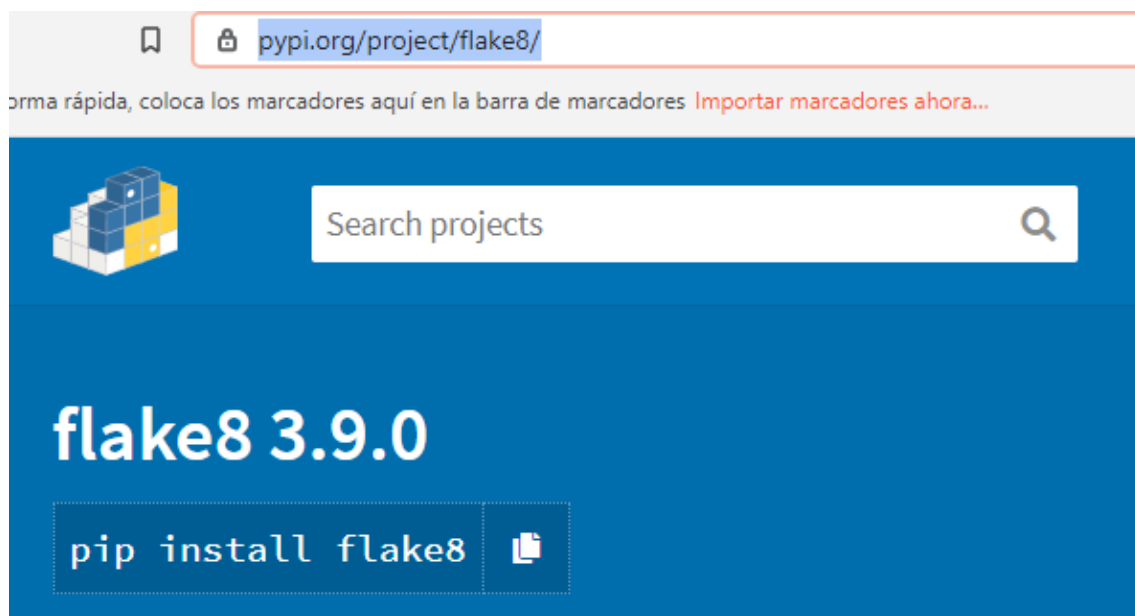


Figura 5.10: Página de la librería flake8

Nos iríamos a la cmd como siempre, y lo que haríamos sería escribirlo.

Trataríamos de instalar nuevamente "linter-flake8" y ya no habría problema.

terminal-plus

El cual no me va correctamente y que es para que me abra la cmd en atom.

atom-terminal-panel

Ahora pruebo este package: y funciona correctamente. Me abre la cmd sin necesidad de ir a la carpeta donde se encuentre el archivo, escribir "cmd" etc.

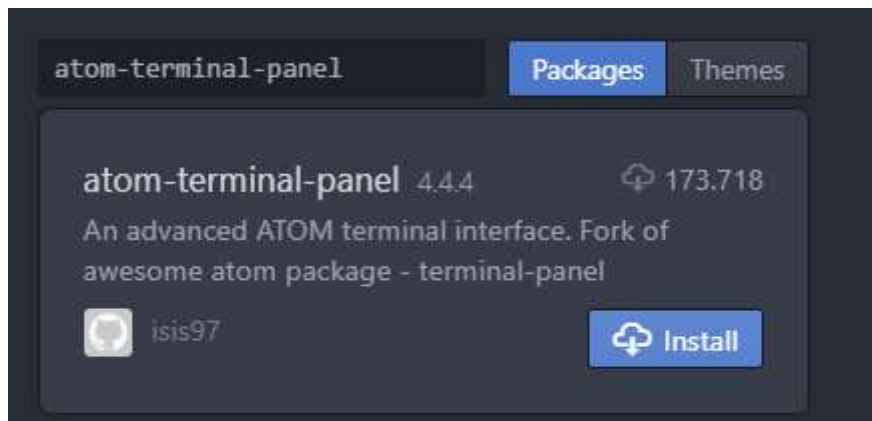


Figura 5.11: Instalación del paquete atom-terminal-panel en Atom

autoclose-html

sirve para cerrar automáticamente etiquetas html una vez abiertas.

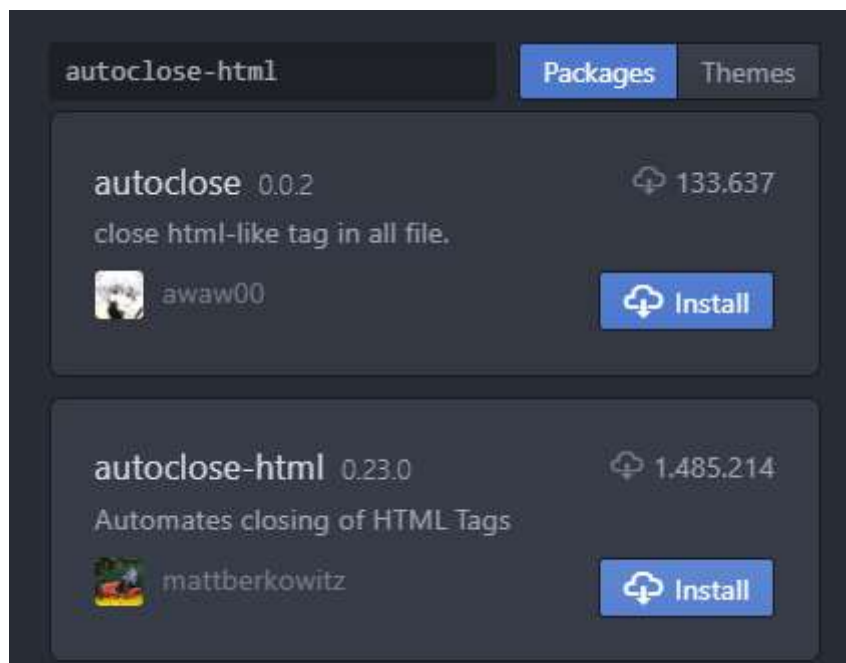


Figura 5.12: Instalación del paquete autoclose-html en Atom

Otros "Packages" interesantes:

(lo mejor es buscar según las necesidades de cada uno/a.

emmet

ideal para desarrolladores frontend

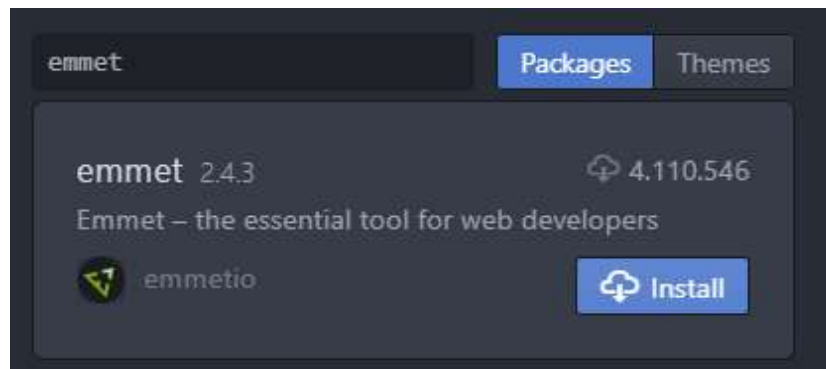


Figura 5.13: Instalación del paquete emmet en Atom

Ask-stack

Para preguntar a Stack Overflow directamente en el IDE.

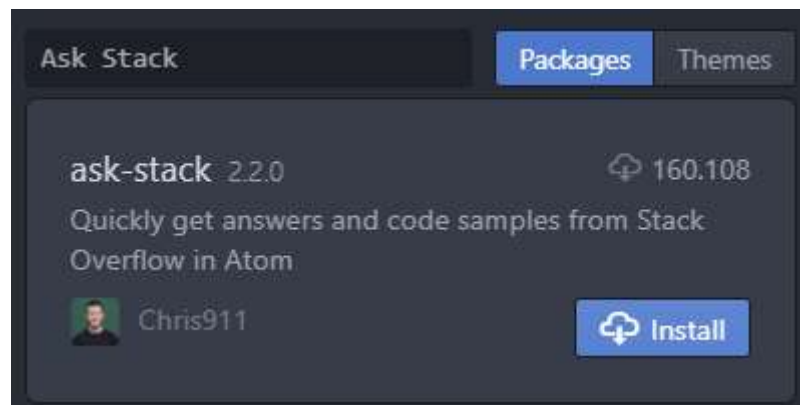


Figura 5.14: Instalación del paquete ask-stack en Atom

Y, tal y como comentamos hay muchos más y dependerá de las necesidades personales de cada desarrollador/a.

6. GUÍA DE INSTALACIÓN POSTMAN PARA WINDOWS

Para ello nos vamos al siguiente Link:

<https://www.postman.com/downloads/>

En mi caso tengo que instalar para 64-bit en Windows.

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

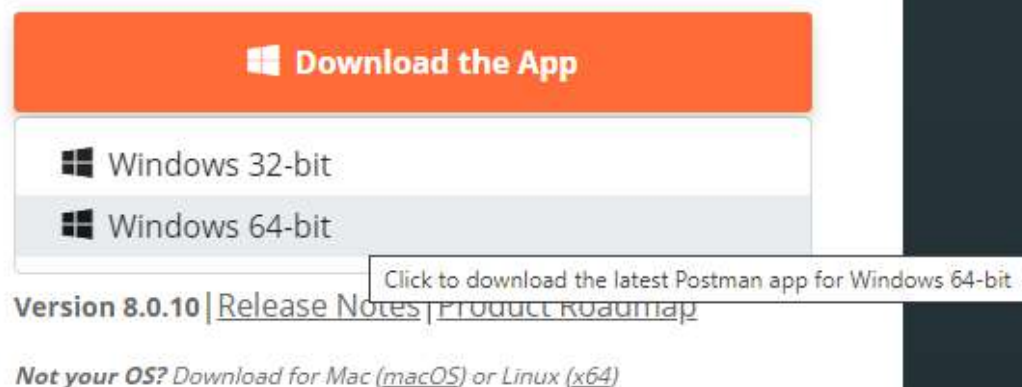
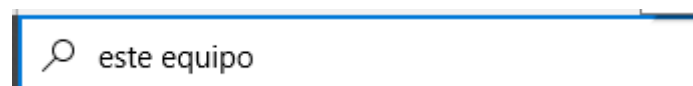


Figura 6.1: Instalación de Postman

Si no sabes que debes instalar puedes escribir: "este equipo en el buscador"

Y te vas a "propiedades".



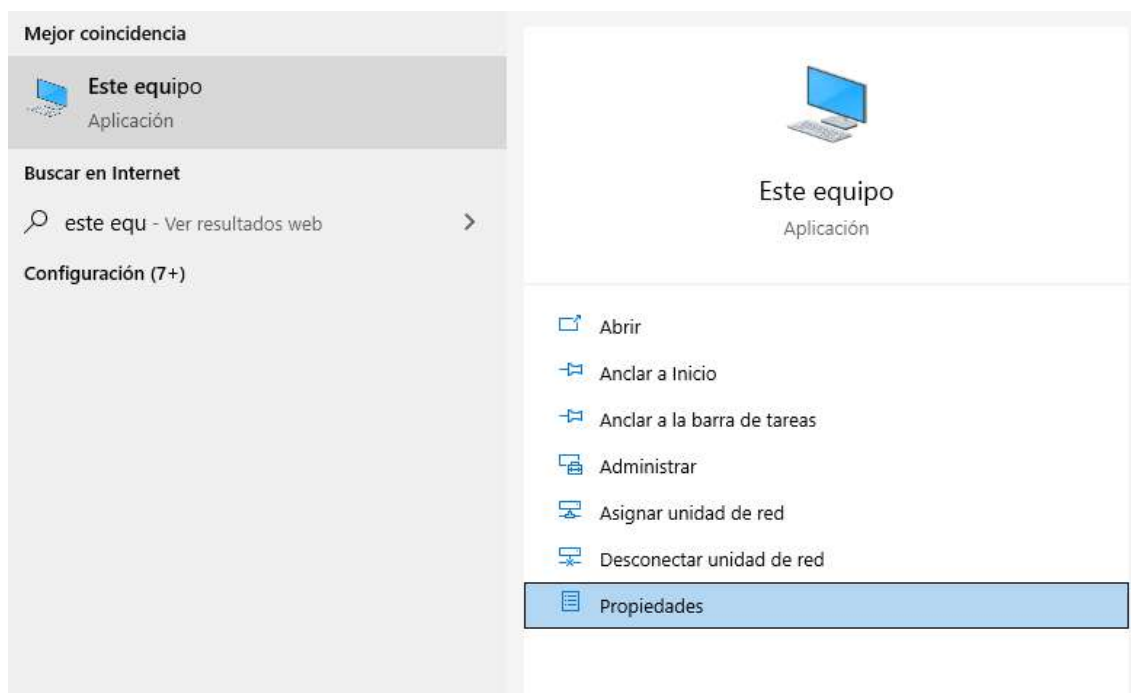


Figura 6.2: Ver propiedades del equipo

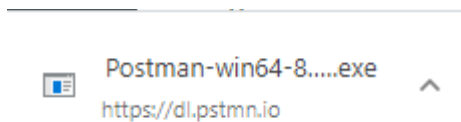
Saldrá algo tal que así:

Sistema		
Fabricante:	Acer	
Modelo:	Aspire ES1-521	
Procesador:	AMD A6-6310 APU with AMD Radeon R4 Graphics	1.80 GHz
Memoria instalada (RAM):	16,0 GB (15,0 GB utilizable)	
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64	
Lápiz y entrada táctil:	Compatibilidad con entrada manuscrita	

Figura 6.3: Ejemplo de propiedades del equipo

Y vemos que dice 64 bits, procesador x64, por lo cual ya sabemos que no debemos instalar la de 32.

Le damos una ruta, y esperamos.



Una vez termina hacemos click sobre ello y nos aparece algo tal que así:



Figura 6.4: Instalación del Postman

Nos dice que puede tardar unos minutos, pero en realidad es muy breve la espera.

Y nos abre lo siguiente:

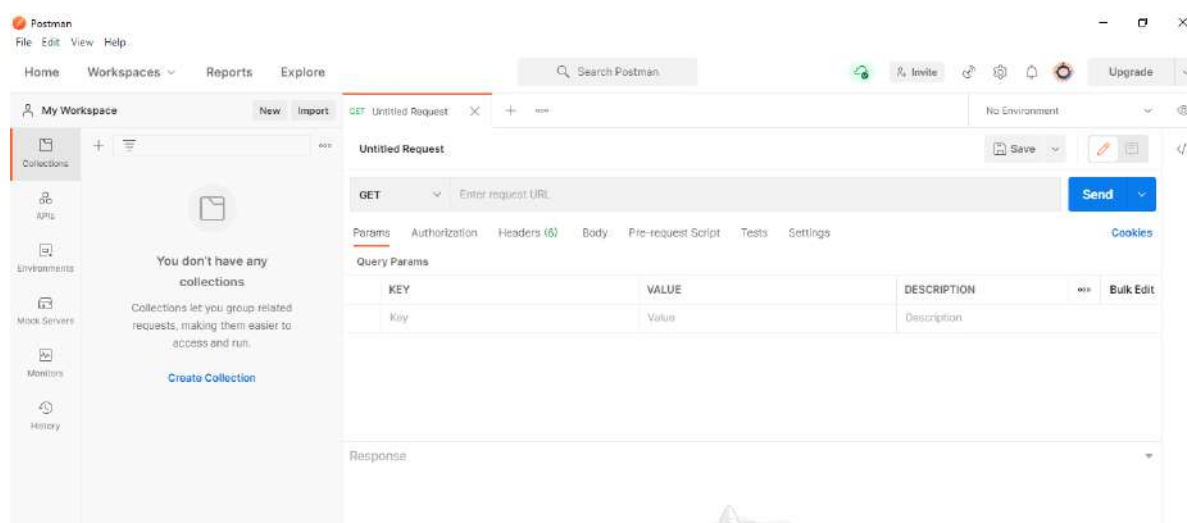


Figura 6.5: Pantalla de inicio de Postman

Ya veremos más adelante para qué sirve este programa.

7. GUÍA DE INSTALACIÓN POSTMAN PARA LINUX UBUNTU 20.04

Nos vamos a las aplicaciones de Linux y buscamos postman:

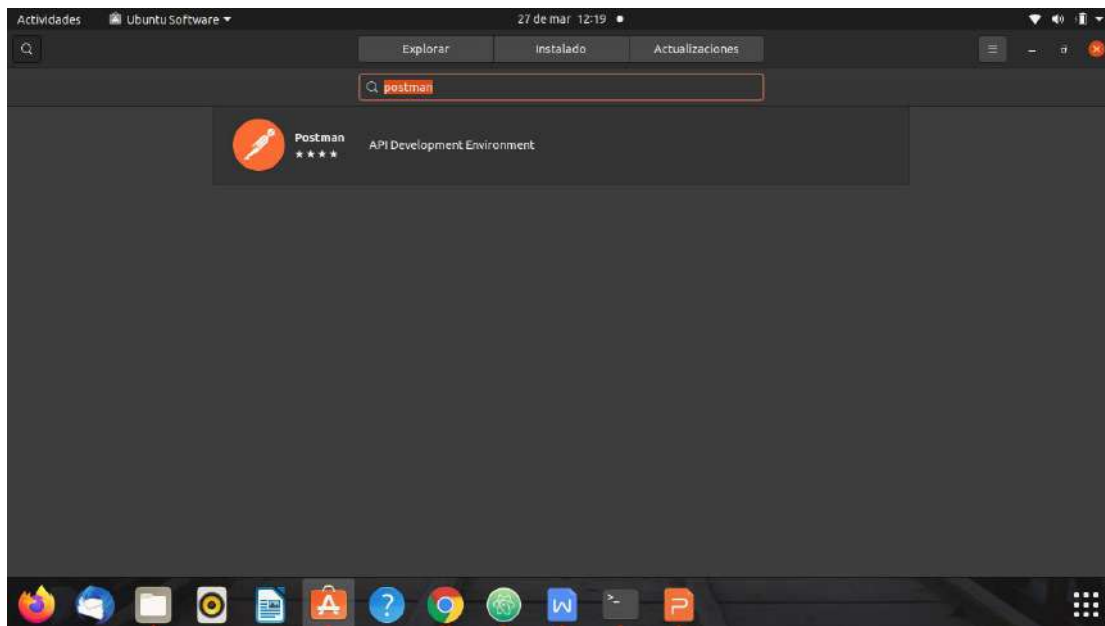


Figura 7.1: Búsqueda de Postman en Ubuntu Software

Hacemos CLICK y pulsamos el botón de INSTALAR:

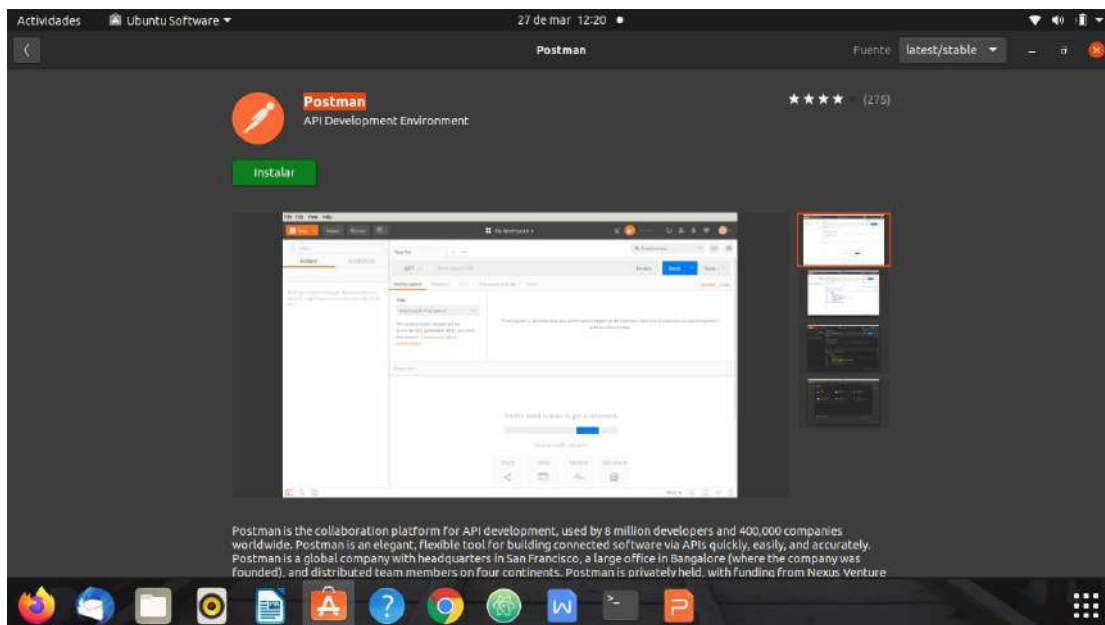


Figura 7.2: Instalación de Postman en Ubuntu Software

Una vez instalado aparecerá en el menú de aplicaciones instaladas:
(esquina inferior derecha)



Figura 7.3: Aplicaciones disponibles en Ubuntu

8. GUÍA DE INSTALACIÓN ENTORNOS VIRTUALES PARA WINDOWS

Existen proyectos en los cuales es necesario disponer de una versión concreta, específica, de una librería, pero en otras ocasiones, en otros proyectos, esa versión de esa librería no funciona correctamente.

De tal manera que para evitar ese problema lo que suele hacerse es usar Entornos Virtuales, de esa forma las versiones de las librerías de diferentes proyectos no entrarán en conflicto.

No es un ejemplo real pero servirá para explicarlo.

Imaginemos que trabajamos en 2 proyectos: Proyecto A y Proyecto B.

El proyecto A necesita una versión 1.16.4 de Numpy, dado que alguna librería, si no cuenta con ella no va a funcionar.

El proyecto B requiere de una versión más reciente, por ejemplo, a mes de abril 2021, 1.19.3 en adelante. (1.20.2 a fecha de 9 abril 2021, la más reciente de Numpy).

La única forma que tenemos de trabajar en ambos proyectos A y B sin que haya problemas será utilizar un Entorno Virtual para el proyecto A y otro Entorno Virtual para el proyecto B.

Por cierto, la forma de instalar esas dependencias sería, por ejemplo:

```
pip install numpy==1.16.4
```

```
pip install numpy==1.19.3
```

Por cierto, tal y como comentamos el ejemplo es inventado totalmente, y Numpy es una librería para realizar operaciones matemáticas, por lo cual existen personas que lo llaman "el equivalente de MATLAB de Python".

Con el mismo crearemos Arrays, Matrices, etc

De modo que comencemos con la instalación!

Instalación del entorno virtual

Nos vamos al "cmd" y escribimos: "pip install virtualenv", y pulsamos Enter

Creación del entorno virtual

Ahora lo que hacemos es crear un entorno virtual:

Nos colocamos en el "cmd" en la carpeta donde queremos el entorno y escribimos:

"virtualenv" + "nombre que le damos al entorno

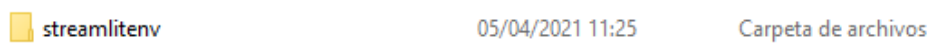
(en nuestro caso lo llamaremos streamlitenv, por ejemplo).

En este ejemplo de prueba:

C:\Users\Manut\Desktop\streamlit>virtualenv streamlitenv

Debería haberse creado una carpeta en esa ruta del PC.

Algo tal que así.



Activación del entorno virtual

Ahora volvemos al "cmd"

Y escribimos: streamlitenv\Scripts\activate

(NOTA: debe ser una barra invertida (backslash) "\" para separarlo)

C:\Users\Manut\Desktop\streamlit>streamlitenv\Scripts\activate

Algo tal que así, donde vemos que ya está activado.

```
C:\Users\Manut\Desktop\streamlit>streamlitenv\Scripts\activate
(streamlitenv) C:\Users\Manut\Desktop\streamlit>
```

Ver librerías instaladas dentro del entorno virtual

Para ver las librerías instaladas en el entorno virtual: "pip list"

```
(streamlitenv) C:\Users\Manut\Desktop\streamlit>pip list
Package      Version
-----
pip          21.0.1
setuptools   54.1.2
wheel        0.36.2
```

Vemos que tendría que instalar una a una cada librería DENTRO de este entorno.

Tendría sentido por tanto, tener diferentes entornos virtuales, aunque debamos instalar nuevamente las librerías.

Desactivar el entorno virtual

Simplemente escribimos: "deactivate" y Enter.

```
(streamlitenv) C:\Users\Manut\Desktop\streamlit>deactivate
C:\Users\Manut\Desktop\streamlit>
```

Ver la lista de librerías con sus versiones



Tendríamos que entrar nuevamente al entorno, por supuesto, y escribir:

pip freeze > requirements.txt

```
(streamlitenv) C:\Users\Manut\Desktop\streamlit>deactivate
C:\Users\Manut\Desktop\streamlit>streamlitenv\Scripts\activate

(streamlitenv) C:\Users\Manut\Desktop\streamlit>pip freeze > requirements.txt
(streamlitenv) C:\Users\Manut\Desktop\streamlit>_
```

Nos iríamos a la carpeta donde se encuentra el entorno virtual, y ahora tendremos un .txt con las versiones de cada librería, en mi caso como aun no tengo instalado nada no tendré nada.

 streamlitenv	05/04/2021 11:25	Carpeta de archivos	
 requirements	05/04/2021 11:57	Documento de te...	0 KB

Ejecutar un Script de Python en el entorno virtual

Para ello, deberíamos activar nuevamente.

Escribimos después: "python" + nombre del script (streamlit_1.py en mi caso)

Y veremos que no tenemos esa librería instalada, entonces "pip install streamlit"

Y en este caso nos instalará un montón de librerías, si escribimos nuestro:

"pip list" nuevamente se verán todas,

Y también en nuestro .txt

```
(streamlitenv) C:\Users\Manut\Desktop\streamlit>deactivate
C:\Users\Manut\Desktop\streamlit>streamlitenv\Scripts\activate

(streamlitenv) C:\Users\Manut\Desktop\streamlit>pip freeze > requirements.txt

(streamlitenv) C:\Users\Manut\Desktop\streamlit>python streamlit_1.py
Traceback (most recent call last):
  File "streamlit_1.py", line 3, in <module>
    import streamlit as st
ModuleNotFoundError: No module named 'streamlit'

(streamlitenv) C:\Users\Manut\Desktop\streamlit>pip install streamlit
Collecting streamlit
  Using cached streamlit-0.79.0-py2.py3-none-any.whl (7.0 MB)
Collecting tzlocal
  Using cached tzlocal-2.1-py2.py3-none-any.whl (16 kB)
Collecting cachetools>=4.0
```

Figura 8.1: Instalación de librerías en entorno virtual

```
(streamlitenv) C:\Users\Manut\Desktop\streamlit>pip freeze > requirements.txt
```

Nuevamente para poder llevarlo al .txt

Y ahora veremos las versiones que tenemos



```
requirements: Bloc de notas
Archivo Edición Formato Ver Ayuda
altair==4.1.0
argon2-cffi==20.1.0
astor==0.8.1
async-generator==1.10
attrs==20.3.0
backcall==0.2.0
base58==2.1.0
bleach==3.3.0
blinker==1.4
cachetools==4.2.1
certifi==2020.12.5
cffi==1.14.5
chardet==4.0.0
click==7.1.2
colorama==0.4.4
decorator==5.0.5
defusedxml==0.7.1
entrypoints==0.3
gitdb==4.0.7
```

Figura 8.2: Ejemplo de requirements.txt

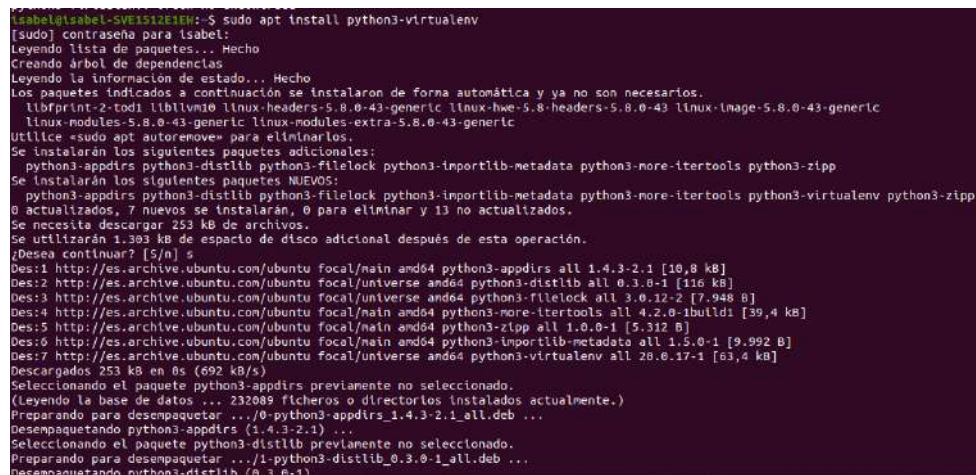
NOTA: Hay muchas más librerías en ese .txt

Por el momento, y para no añadirle más complejidad, no veremos como pasar estas librerías a otro entorno virtual, dado que es algo que no utilizaremos.

9. GUÍA DE INSTALACIÓN ENTORNOS VIRTUALES PARA LINUX UBUNTU 20.04

Para generar nuestro entorno virtual en el que debemos trabajar instalaremos virtualenv para evitar problemas entre las versión de las librerías:

```
pip install virtualenv or sudo apt install python3-virtualenv
```



```
isabel@isabel-SVE1512E1H:~$ sudo apt install python3-virtualenv
[sudo] contraseña para isabel:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libfprint-2-tod1 libblvm10 linux-headers-5.8.0-43-generic linux-hwe-5.8-headers-5.8.0-43 linux-image-5.8.0-43-generic
  linux-modules-5.8.0-43-generic linux-modules-extra-5.8.0-43-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  python3-appdirs python3-distlib python3-filelock python3-importlib-metadata python3-more-itertools python3-zipp
Se instalarán los siguientes paquetes NUEVOS:
  python3-appdirs python3-distlib python3-filelock python3-importlib-metadata python3-more-itertools python3-virtualenv python3-zipp
0 actualizados, 7 nuevos se instalarán, 0 para eliminar y 13 no actualizados.
Se necesita descargar 253 kB de archivos.
Se utilizarán 1.303 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [5/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu focal/main amd64 python3-appdirs all 1.4.3-2.1 [10,8 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 python3-distlib all 0.3.0-1 [116 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 python3-filelock all 3.0.12-2 [7.948 B]
Des:4 http://es.archive.ubuntu.com/ubuntu focal/main amd64 python3-more-itertools all 4.2.0-1build1 [39,4 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu focal/main amd64 python3-zipp all 1.0.0-1 [5.312 B]
Des:6 http://es.archive.ubuntu.com/ubuntu focal/main amd64 python3-importlib-metadata all 1.5.0-1 [9.992 B]
Des:7 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 python3-virtualenv all 20.0.17-1 [63,4 kB]
Descargados 253 kB en 0s (692 kB/s)
Seleccionando el paquete python3-appdirs previamente no seleccionado.
(Leyendo la base de datos ... 232089 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../0-python3-appdirs_1.4.3-2.1_all.deb ...
Desempaquetando python3-appdirs (1.4.3-2.1) ...
Seleccionando el paquete python3-distlib previamente no seleccionado.
Preparando para desempaquetar .../1-python3-distlib_0.3.0-1_all.deb ...
Desempaquetando python3-distlib (0.3.0-1)
```

Figura 9.1: Instalación de virtualenv en Ubuntu

Creamos el entorno virtual mediante el siguiente comando:

```
virtualenv <nombreEntorno>
```

```
virtualenv my_django_app
```

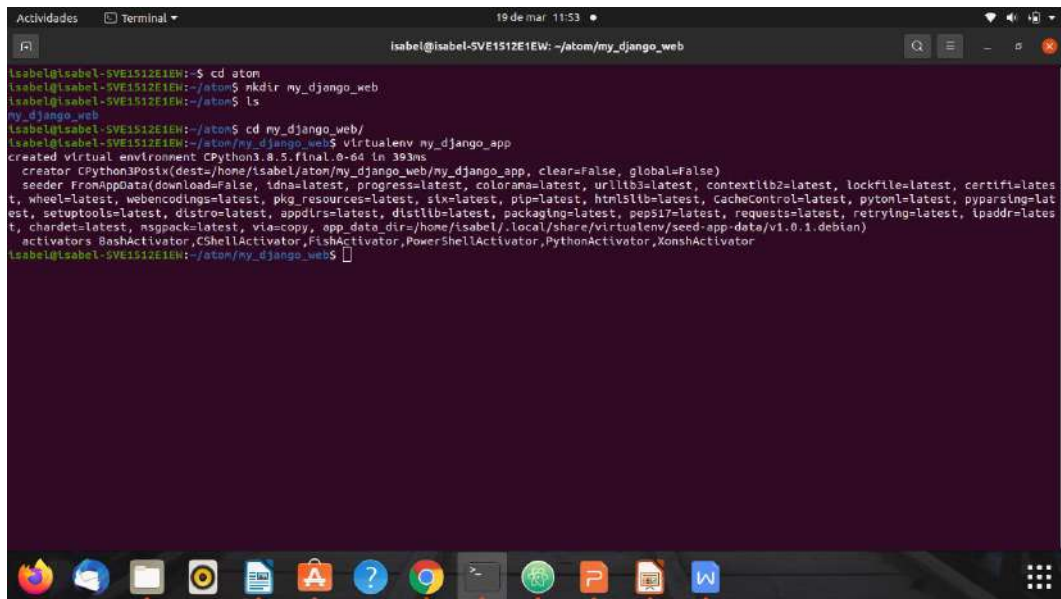


Figura 9.2: Creación del Entorno Virtual en Ubuntu

Observamos que en nuestra carpeta creada my_django_web se nos ha creado otra con el entorno virtual con el nombre my_django_app:

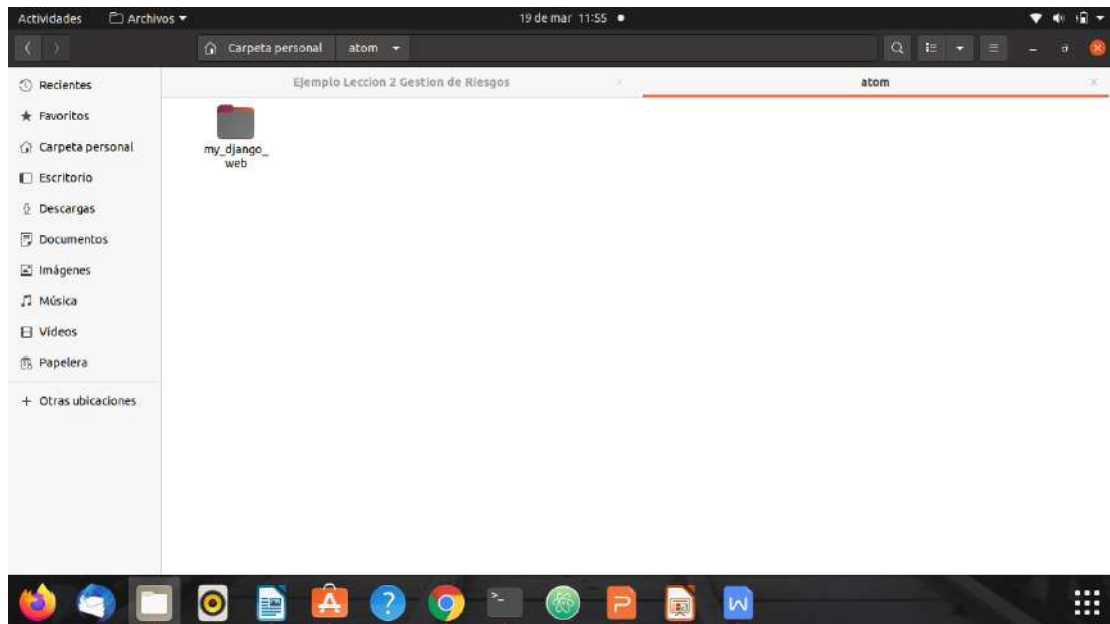


Figura 9.3: Creación del Entorno Virtual en Ubuntu

Una vez creado accedemos al entorno mediante:

```
source my_django_app/bin/activate
```

```

isabel@isabel-SVE1512E1EW:~$ cd atom
isabel@isabel-SVE1512E1EW:~/atom$ mkdir my_django_web
isabel@isabel-SVE1512E1EW:~/atom$ cd my_django_web
isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ virtualenv my_django_app
Created virtual environment CPython3.8.5.final.0-64 in 393ms
creator CPython3Posix(dest=/home/isabel/atom/my_django_web/my_django_app, clear=False, global=False)
seeder FromAppData(download=False, idna=latest, progress=latest, colorama=latest, urllib3=latest, contextlib2=latest, lockfile=latest, certifi=late
t, wheel=latest, webcodings=latest, pkg_resources=latest, six=latest, pip=latest, html5lib=latest, cachecontrol=latest, pytoml=latest, pyparsing=late
st, setuptools=latest, distro=latest, appdirs=latest, distlib=latest, packaging=latest, pep517=latest, requests=latest, retrying=latest, ipaddr=late
t, chardet=latest, msgpack=latest, via=copy, app_data_dir=/home/isabel/.local/share/virtualenv/seed-app-data/v1.0.1.debian)
activators BashActivator,CSHELLActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ source my_django_app/bin/activate
(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web$
  
```

Figura 9.4: Activación del Entorno Virtual en Ubuntu

Vemos que entre paréntesis se encuentra nuestro Entorno Virtual creado my_django_app.

En dicho entorno podemos chequear las librerías instaladas y en la versión en la que se encuentra así podremos usar distintos entornos sin afectarnos unos a otros:

```
pip list
```

```

(my_django_app) isabel@isabel-SVE1512E1EW:~/atom/my_django_web$ pip list
Package            Version
-----
appdirs            1.4.3
CacheControl       0.12.6
certifi            2019.11.28
chardet            3.0.4
colorama           0.4.3
contextlib2        0.6.0
distlib            0.3.0
distro             1.4.0
html5lib           1.0.1
idna               2.8
ipaddr             2.2.0
lockfile           0.12.2
msgpack            0.6.2
packaging          20.3
pep517             0.8.2
pip               20.0.2
pkg_resources      0.0.0
progress           1.5
pyparsing          2.4.6
pytoml             0.1.21
requests           2.22.0
retrying           1.3.3
setuptools         44.0.0
six               1.14.0
urllib3            1.25.8
webcodings         0.5.1
wheel              0.34.2
  
```

Figura 9.5: Librerías disponible en el Entorno Virtual en Ubuntu

Para cerrar el Entorno Virtual usaremos:

```
deactivate
```

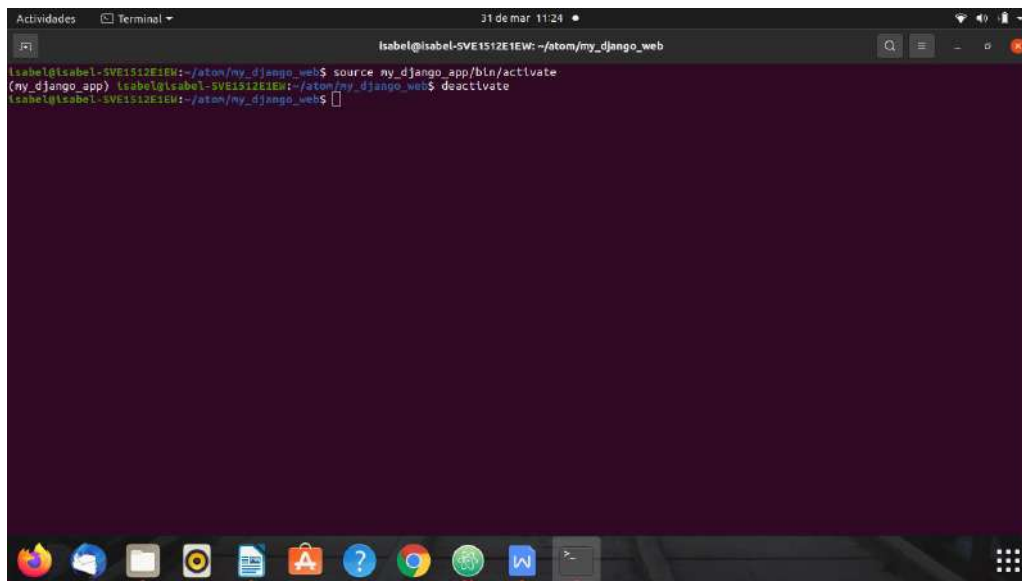


Figura 9.6: Desactivar el Entorno Virtual en Ubuntu

Observamos que se nos ha cerrado el entorno ya que no aparece entre paréntesis el entorno.



Recuerda

Nuestra aplicación siempre debe tener un Entorno Virtual para evitar problemas con las versiones de las distintas librerías.

10. ANACONDA: NO NECESARIA INSTALACIÓN PERO UTILIZADO PUNTUALMENTE.

Instalación NO Necesaria, VOLUNTARIA- aunque recomendable en el futuro.

Dentro de la Inteligencia Artificial existen varios Entornos de Desarrollo.

Uno de ellos es muy popular en proyectos en los cuales existe necesidad de hacer bastantes gráficas, y algo que se conoce como "*data mining*", que no tiene relevancia en la presente asignatura. Este Entorno, "Jupyter" es algo que usan empresas muy importantes a nivel mundial y el mismo podemos usarlo al instalar Anaconda, por ejemplo.

¿Por qué entonces instalaríamos Anaconda?

El motivo es que al instalar Anaconda estamos preinstalando a su vez mas Entornos de Desarrollo, y muchísimas librerías científicas.

A año 2021, no obstante, se suele afirmar que aunque este entorno es muy interesante, "Google Colab" es incluso mejor elección para proyectos de Data Science.

PyCharm, y el propio ATOM, son otras 2 opciones.

No incluiremos la guía, pero es un Software que dispone de varios IDE, uno de ellos es Jupyter, y el cual utilizaremos en 2 momentos puntuales del curso.

Primeramente para hacer un breve resumen de Python sobre el cual veamos paso a paso como se ejecuta el código.

Y en uno de los Frameworks existe una opción sobre este entorno.

El link de donde podríamos instalar Anaconda es el siguiente:

<https://www.anaconda.com/>

O si queremos instalar solamente el entorno Jupyter:

De esta web:

<https://jupyter.org/install>

en donde encontraríamos lo siguiente: "pip install notebook"

11. GUÍA DE INSTALACIÓN QT DESIGNER EN WINDOWS

Para ello nos vamos al siguiente Link:

<https://build-system.fman.io/qt-designer-download>

(Windows y Mac)



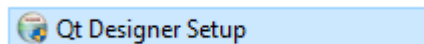
Many people want to use Qt Designer without having to download gigabytes of other software. Here are small, standalone installers of Qt Designer for Windows and Mac:



Figura 11.1: Instalación de QT-Designer en Windows

Elegimos nuestro sistema operativo

Una vez ahí doble click:



Y nos encontramos:

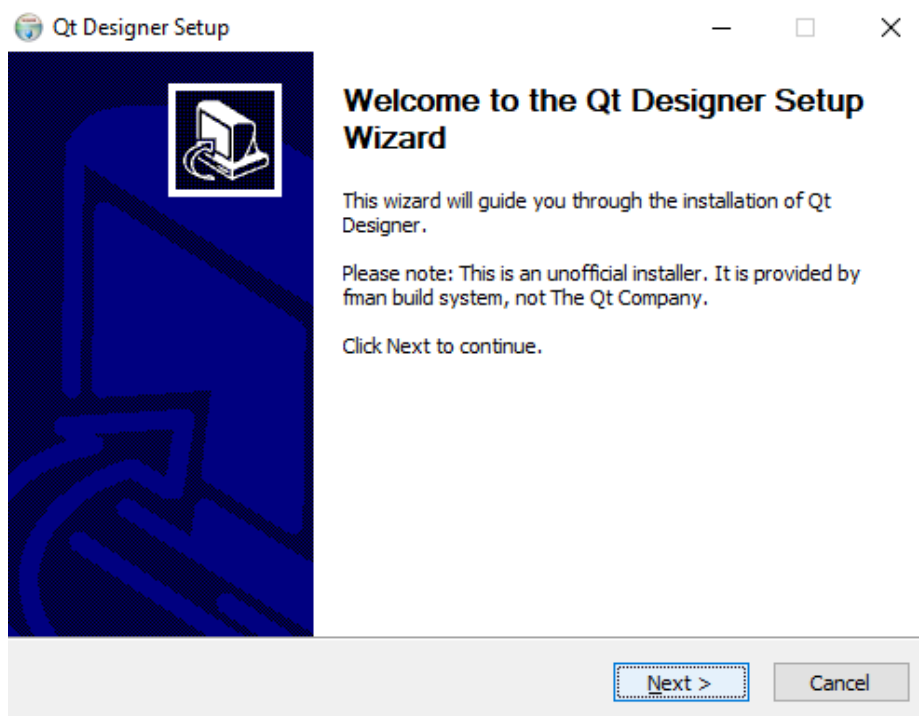


Figura 11.2: Instalación de QT-Designer wizard en Windows

Next→ Ahora elegimos una ruta e instalamos.

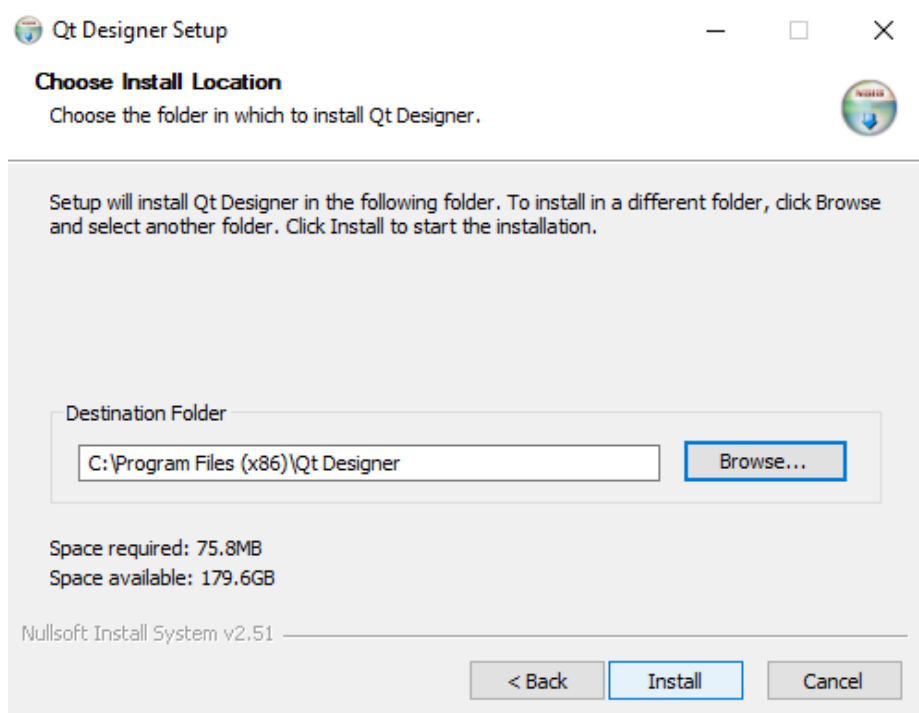


Figura 11.3: Instalación de QT-Designer wizard en Windows

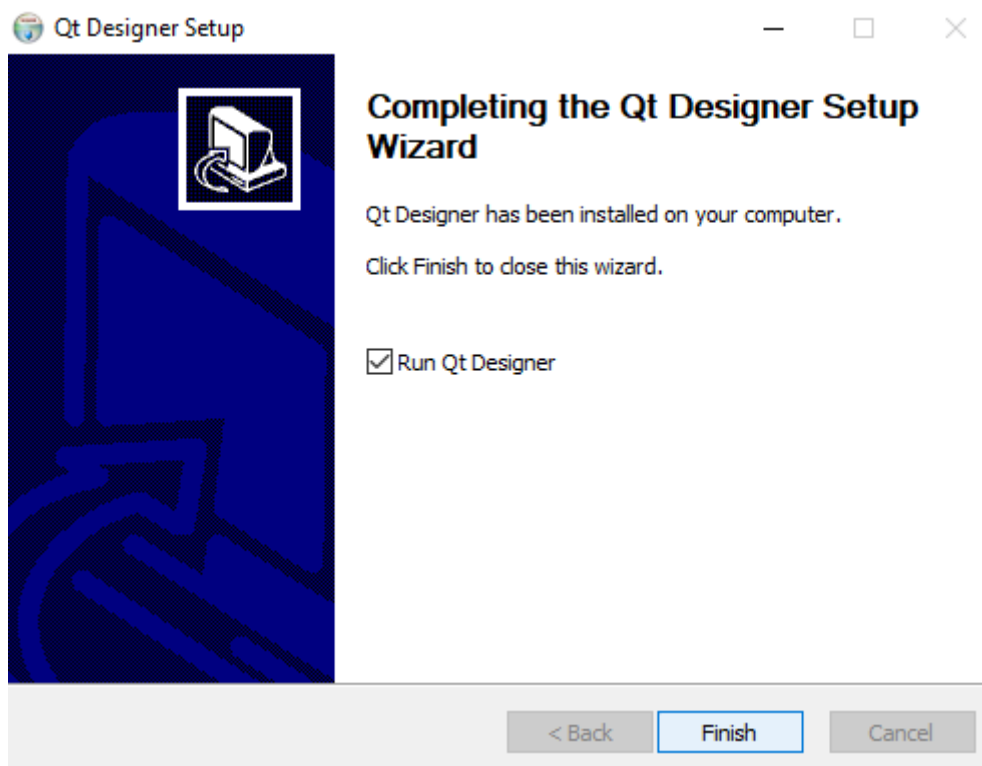


Figura 11.4: Instalación de QT-Designer wizard en Windows

Y tenemos el Qt Designer instalado en pocos minutos

Con una apariencia como esta.

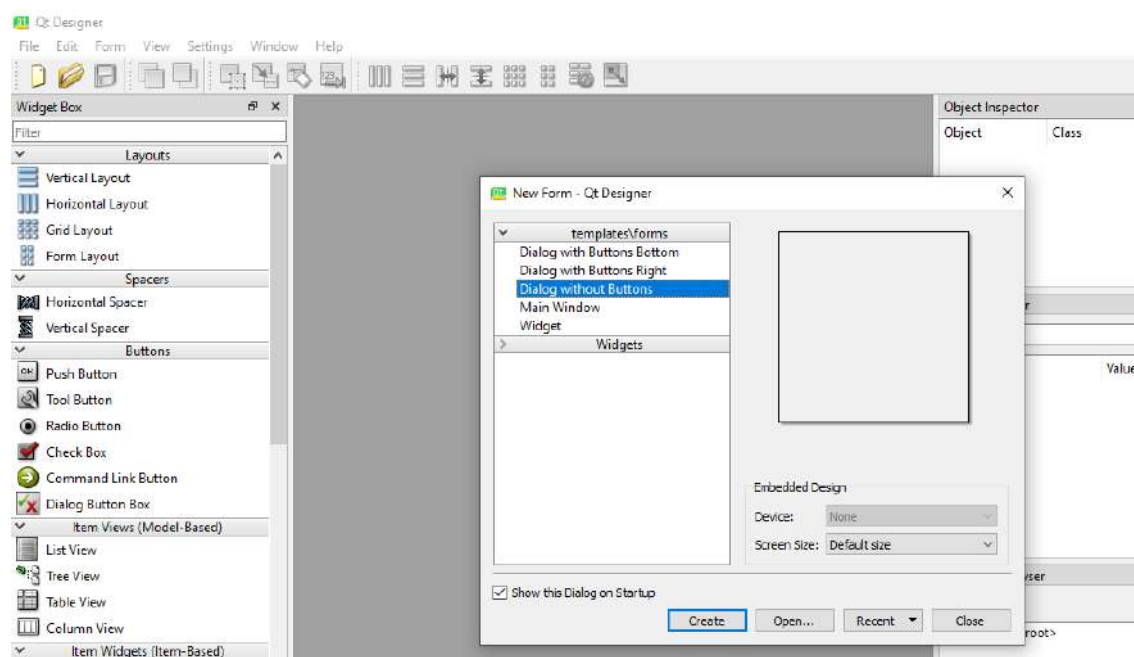


Figura 11.5: Pantalla inicial de QT-Designer en Windows

Después vamos a la carpeta donde está la aplicación:

`cd /usr/lib/x86_64-linux-gnu/qt5/bin/` y escribo: "designer"

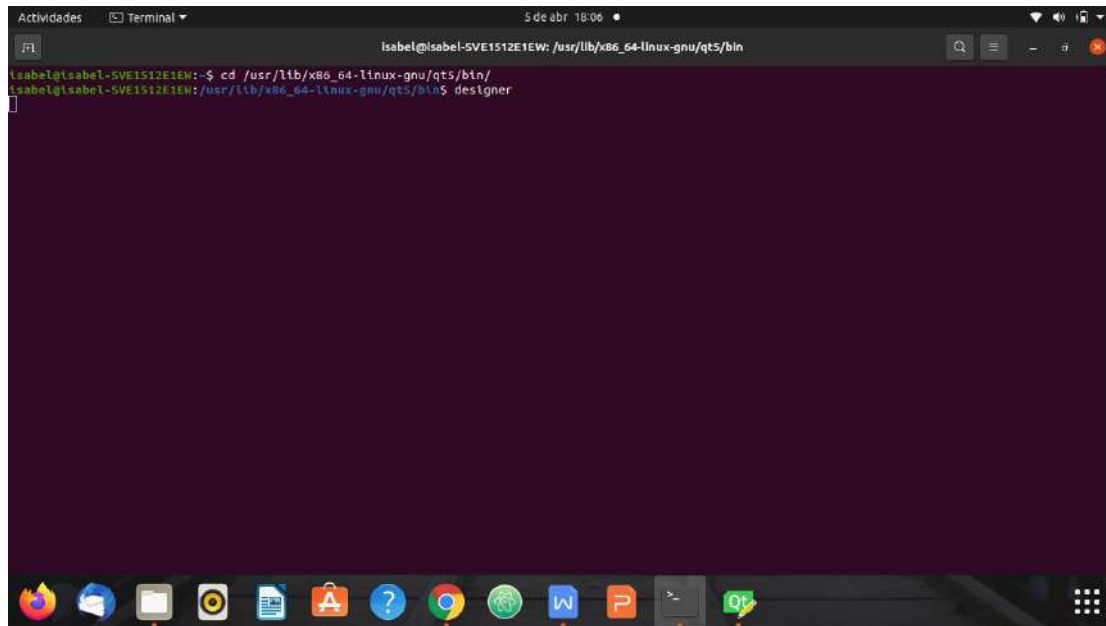


Figura 12.3: Instalación de QT-Designer en Ubuntu

Nos abre la aplicación:

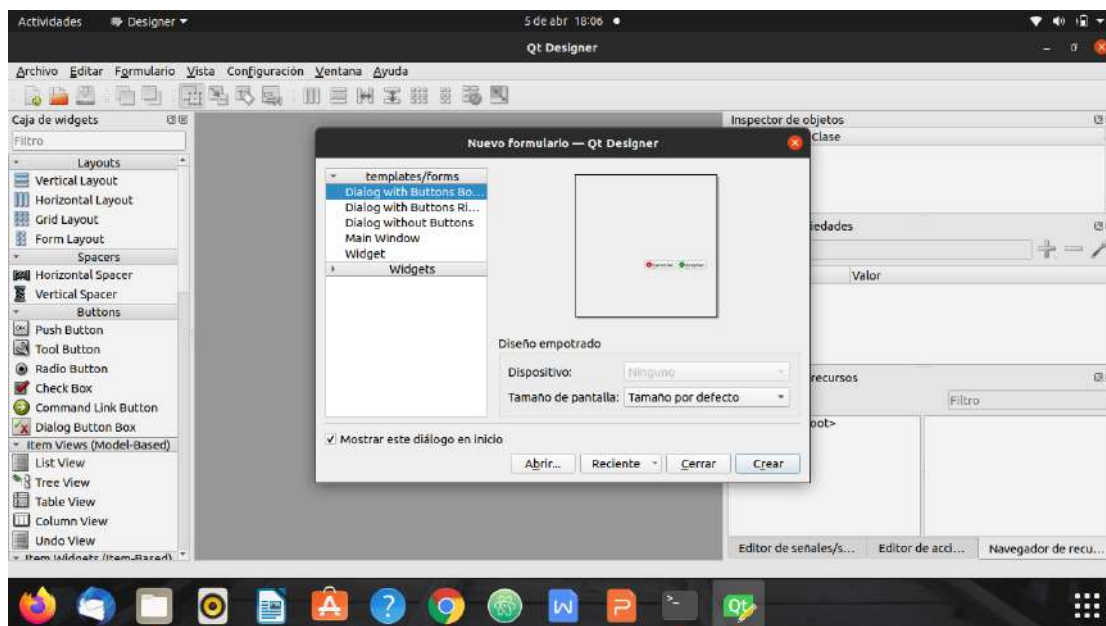


Figura 12.4: Pantalla inicial de QT-Designer en Ubuntu

13. PUNTOS CLAVE

- | Python se está convirtiendo en el Lenguaje de Programación de referencia y cada día disponemos de más librerías y de más recursos.
- | ATOM es un buen Entorno de Desarrollo, que cada día usan más desarrolladores y el cual será importante tenerlo instalado previo al tema donde se hablará de Django, para poder seguir las explicaciones.
- | Postman es necesario para esta Asignatura/Módulo, por lo que será necesario tenerlo instalado. Será utilizado en varias ocasiones.
- | Resulta conveniente aprender qué es un Entorno Virtual y su utilidad en aquellos momentos en los que trabajamos en varios proyectos al mismo tiempo.

